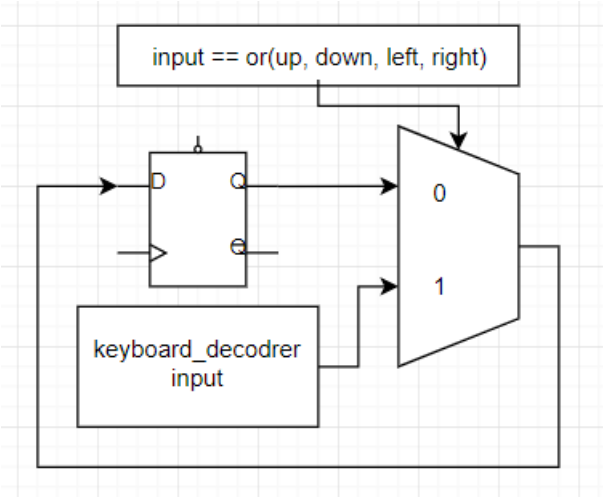
Q1 VGA_keyboard

keyboard 訊號處理

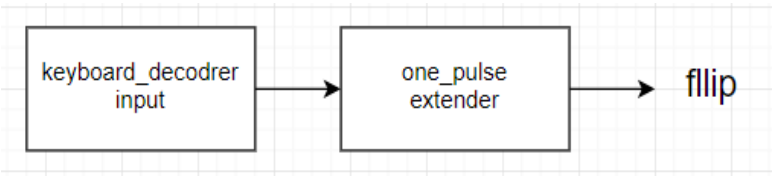
[2:0] key :

key為3-bit register，用來儲存鍵盤上、下、左、右的輸入，當鍵盤輸一個按鍵時，key的值會儲存該值直到下一個鍵輸入。下邏輯電路圖如下 :



h_flip, v_flip, pause :

由於flip的訊號只能在一個週期內被讀取，因此要在輸入時會將pause, h_flip, v_flip的訊號轉為one pulse訊號，但由於keyboard decoder所製成的one pulse訊號不夠長，沒辦法在一個vga display週期內都維持住，因此需要透過加長的機制將one pulse extend，如此一來，才能夠確保訊號能夠讀進去。此部分的block diagram如下。



螢幕position 控制

控制螢幕位置的偏移量由position所控制，原本的位置皆會加上position，當加上position位置大於垂直邊界時，會取餘數，大於垂直邊界部分的圖片會跑到最上面，而此次的Lab藉由改變position的數值來達成螢幕水平、垂直平移的效果。

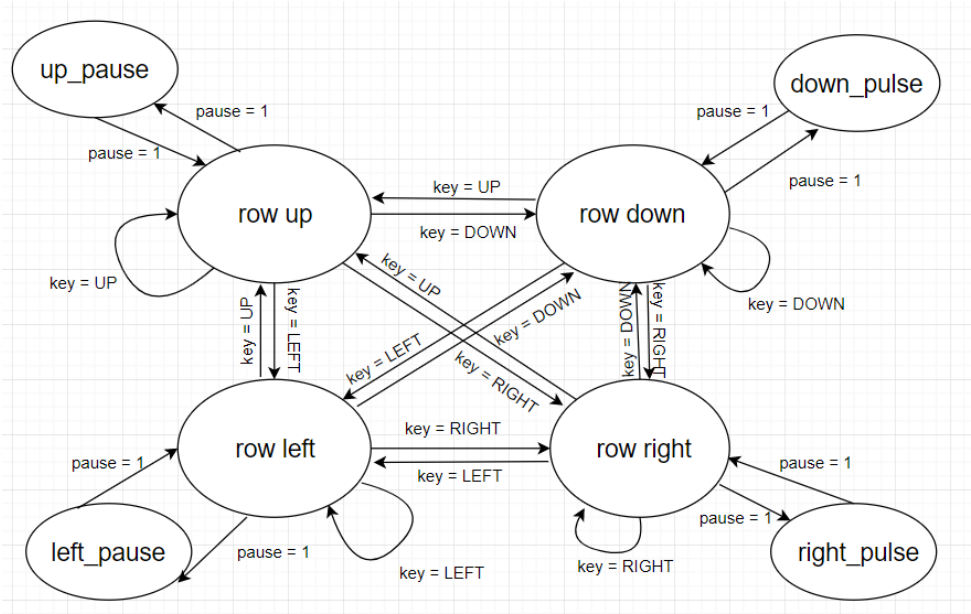
1. 上下左右平移

控制上下左右及暫停的功能藉由 Finite-State-Machine 的概念完成，我們定義了八個State，分別為上移、下移、左移、右移及四個移動過程中暫停的功能。
當rst == 1’ b1 時，圖片靜止，預設是停留在 State 0。

State up_pause為上移過程的暫停，讀取到P訊號時會進入State row_up，繼續上移。
State down_pause為下移過程的暫停，讀取到P訊號時會進入State row_down，繼續下移。
State left_pause為下移過程的暫停，讀取到P訊號時會進入State row_left，繼續下移。
State right_pause為下移過程的暫停，讀取到P訊號時會進入State row_right，繼續下移。

State row_up為上移的state，當讀取到下、左、右時，分別會進入到row_down, row_left, row_right的state 繼續移動，讀取到P訊號時會進入up_pause停止不動。
State row_down為下移的state，當讀取到上、左、右時，分別會進入到row_up, row_left, row_right的state 繼續移動，讀取到P訊號時會進入down_pause停止不動。
State row_left為左移的state，當讀取到下、上、右時，分別會進入到row_up, row_left, row_right的state 繼續移動，讀取到P訊號時會進入left_pause停止不動。
State row_right為右移的state，當讀取到下、左、上時，分別會進入到row_down, row_left, row_up的state 繼續移動，讀取到P訊號時會進入right_pause停止不動。

State diagram 如下 :



2. 水平垂直翻轉

完成移動功能之後，接著製作翻轉功能，為了達成上下、左右間的切換，加入了direction_x, direction_y作為判斷。
原本pixel_addr 輸出的式子為 :

(320*((v_cnt>>1)+position_y)+((h_cnt>>1)+position_x)%320)% 76800，

但為了使輸出的圖片上下相反，我們將式子改為 :

(320*((v_cnt>>1)+position_y)+320-((h_cnt>>1)+position_x)%320)% 76800 ，

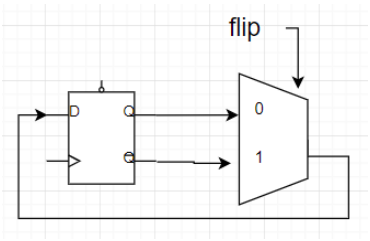
取了一個負號之後，上下的方向就能夠被翻轉，
同理，左右翻轉的式子為 :

(320*(240-((v_cnt>>1)+position_y)%240)+((h_cnt>>1)+position_x)%320)% 76800。

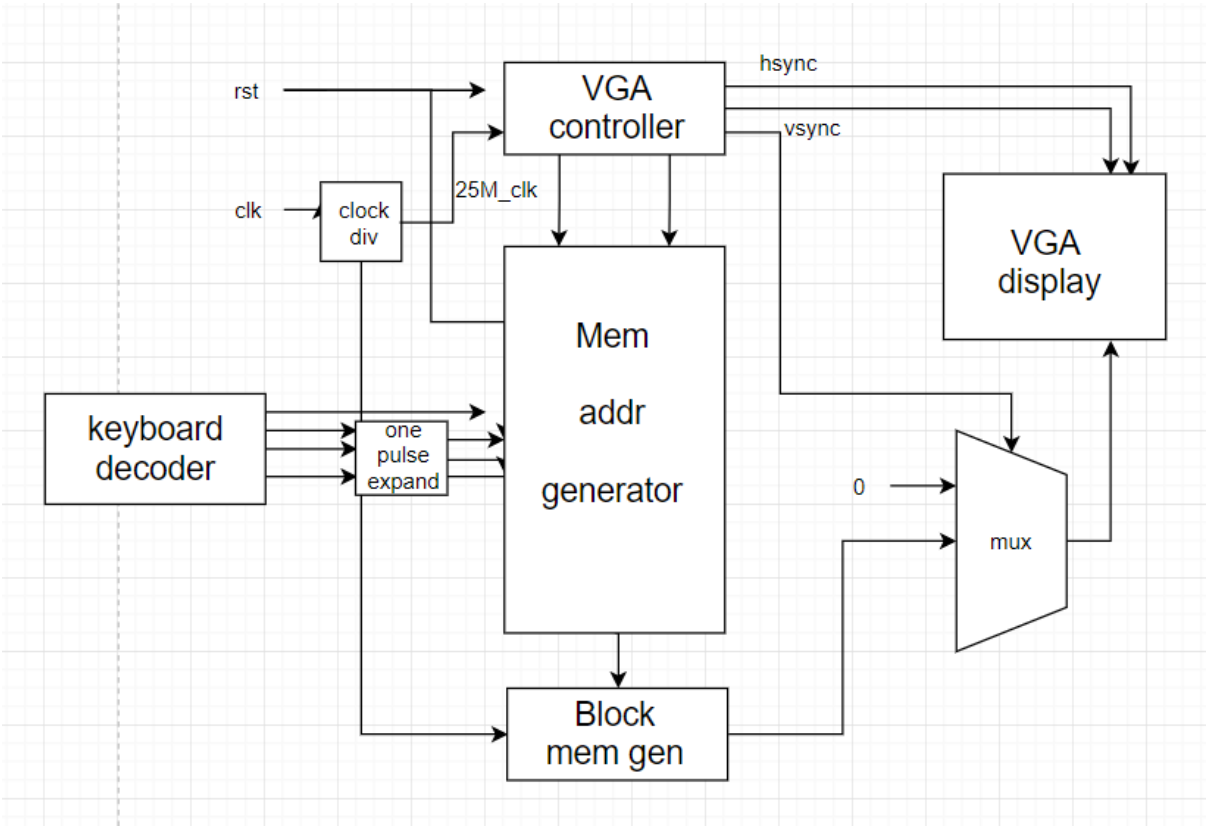
上下左右接相反的式子為 :

76800-((320*((v_cnt>>1)+position_y)+((h_cnt>>1)+position_x)%320)% 76800)

在rst階段時，direction_x被設為1’ b0，direction_y為1’ b0，當h_flip = 1’ b1時，direction_x會取負號，而當v_flip = 1’ b1時，direction_y會取負號，以下為flip 與 direction 的邏輯電路。



整體Block Diagram 如下



Verification

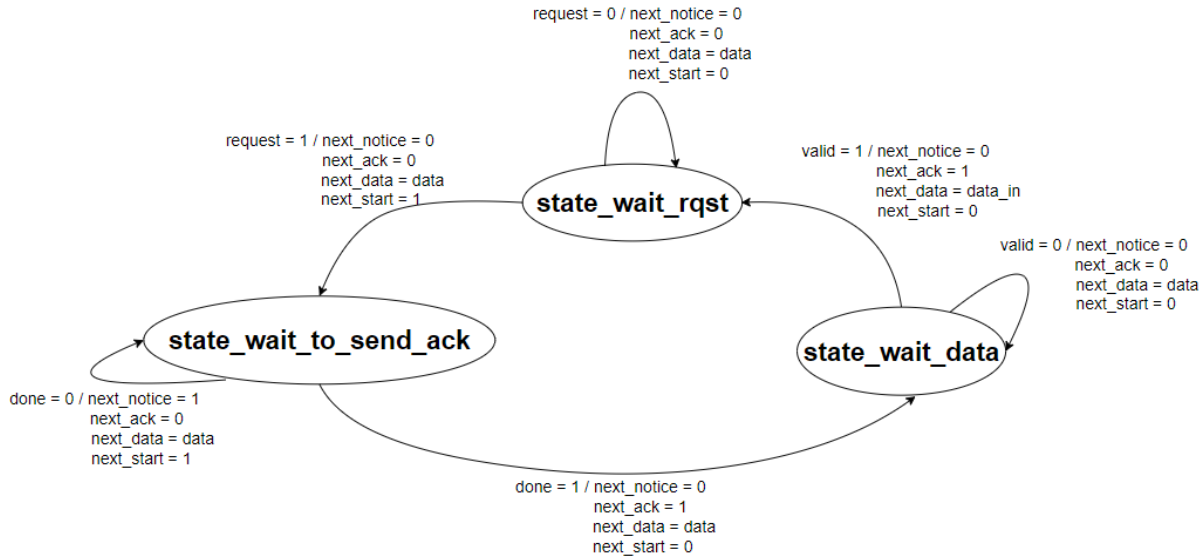
當上移時，能夠切換到下、左、右，能切到暫停且切得回來，能水平、垂直翻轉。
當下移時，能夠切換到上、左、右，能切到暫停且切得回來，能水平、垂直翻轉。
當左移時，能夠切換到下、上、右，能切到暫停且切得回來，能水平、垂直翻轉。
當右移時，能夠切換到下、左、上，能切到暫停且切得回來，能水平、垂直翻轉。
按下按鈕時能夠回到初始狀況。

綜合以上，能夠確認此份程式碼為正確。

Dual FPGA communication

(一)State Transition Diagram

以下是slave_control的State Transition Diagram：

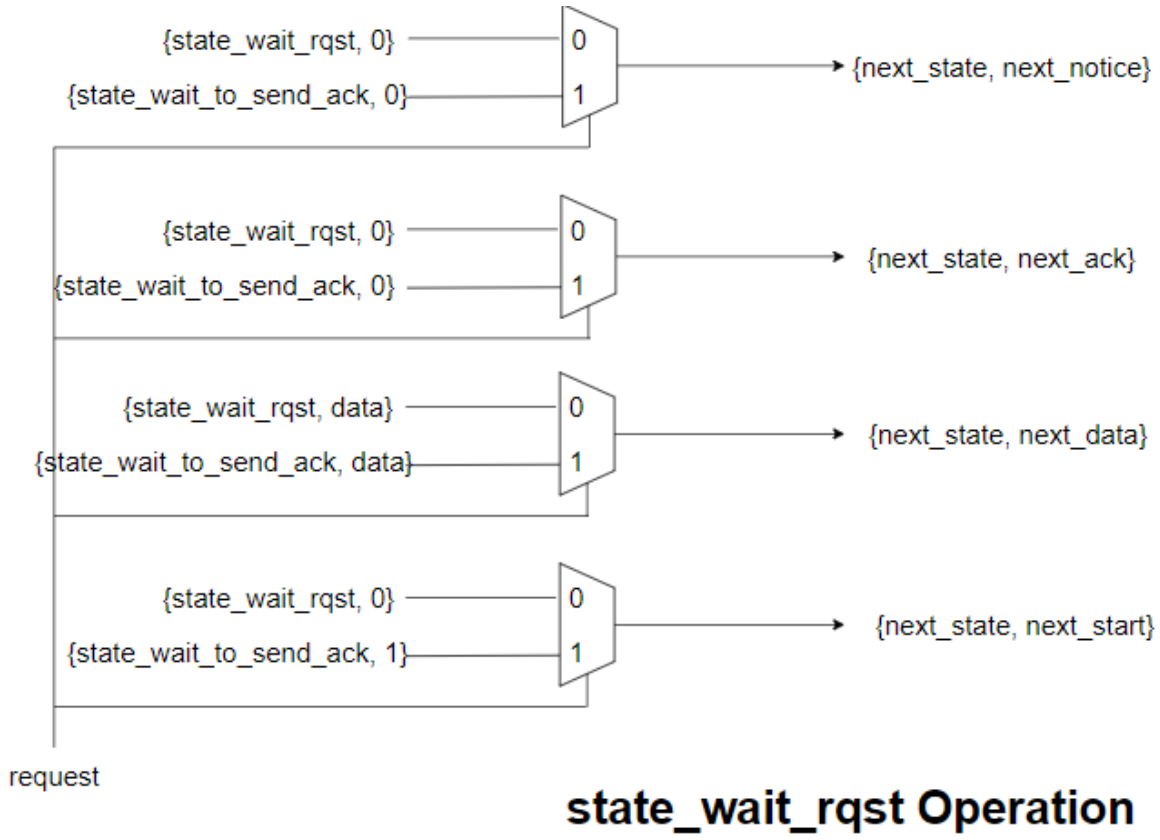


(二)Circuit Design and explanation：

#(notice訊號是控制LED燈是否為亮的訊號,data訊號是七段顯示器上顯示的數字,start訊號是counter開始計數的訊號(因為LED燈要亮>=1s),done訊號是表示conter是否數到1秒的訊號,valid訊號是從master端傳來的，是要告訴我們data_in已經傳送到slave端了)

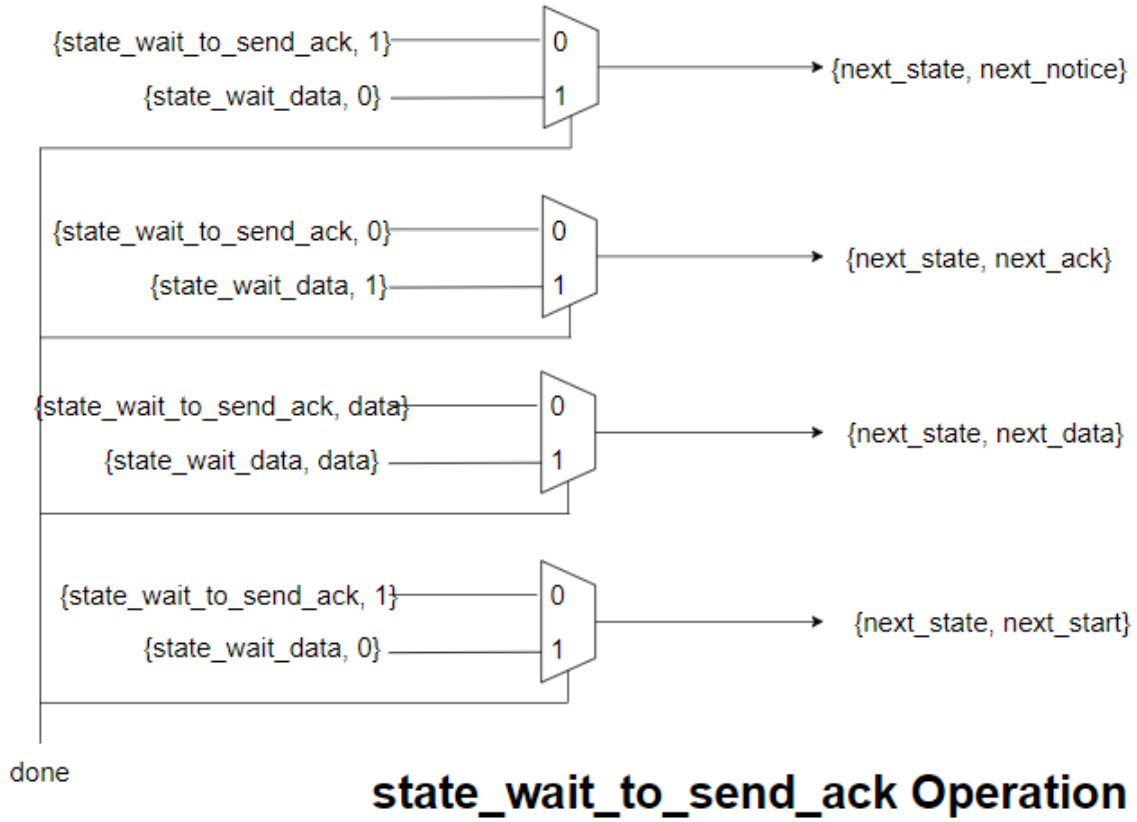
1.當一開始slave端按下rst_n時，slave端的七段顯示器會歸零，並進入state_wait_rqst的state。

2.進入state_wait_rqst後，這個state是在等待master端的request的訊號，當收到request的訊號時(namely request = 1’ b1)，會進入到state_wait_to_send_ack，然後next_notice = 1’ b0, next_ack = 1’ b0, next_data = data, next_start = 1’ b1；反之，如果沒有收到request的訊號，會一直維持在state_wait_rqst這個state，next_notice = 1’ b0, next_ack = 1’ b0, next_data = data, next_start = 1’ b0。



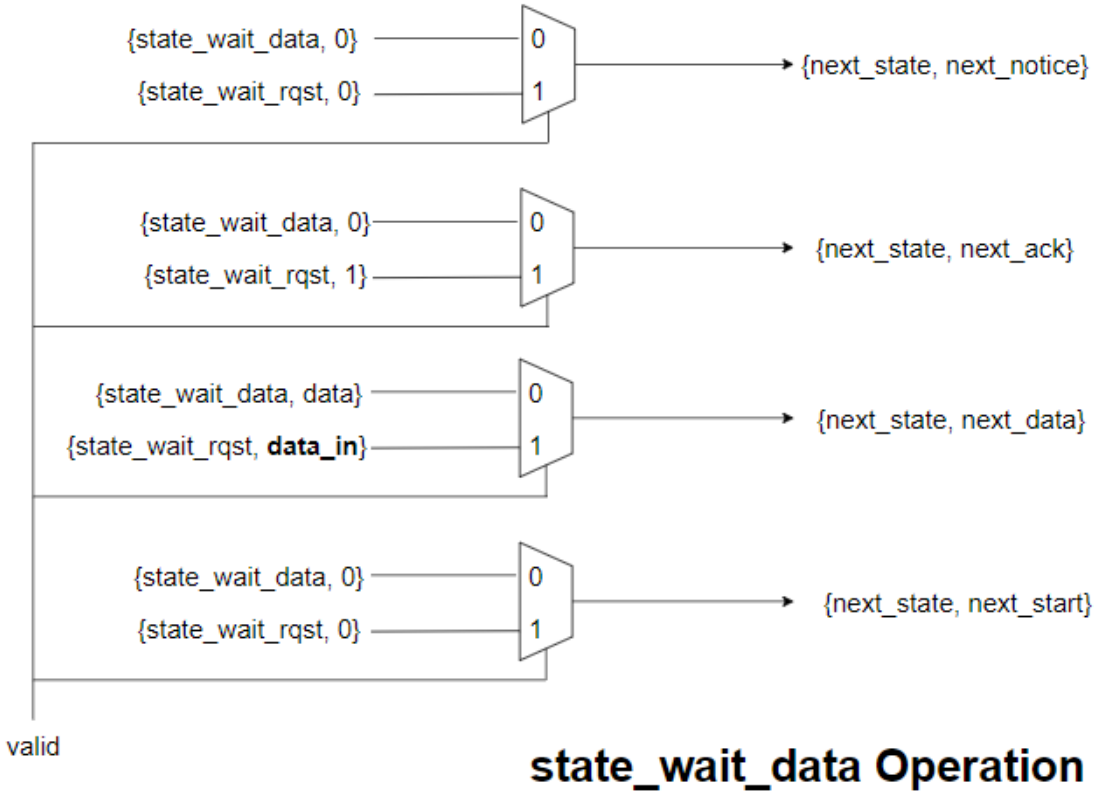
state_wait_rqst Operation

3.進入state_wait_to_send_ack後，這個state是要讓slave端的LED燈亮1秒後(LED燈亮表示有收到master端request的訊號)，接著要傳送ack的訊號給master端。所以當done = 1’ b1時，代表已經維持了1秒了，接著會進入到state_wait_data，然後next_notice = 1’ b0, next_ack = 1’ b1, next_data = data, next_start = 1’ b0；反之，如果當done = 1’ b0時，會一直維持在state_wait_to_send_ack這個state，next_notice = 1’ b1, next_ack = 1’ b0, next_data = data, next_start = 1’ b1。

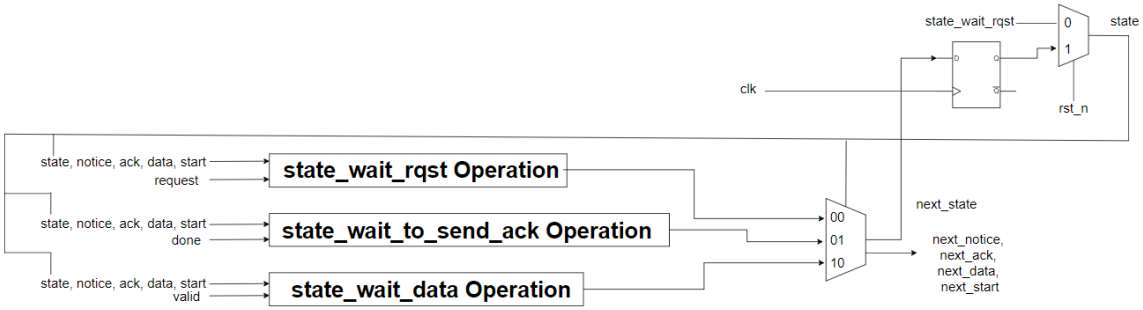


state_wait_to_send_ack Operation

4.進入state_wait_data後，這個state主要是要確認slave端是否有收到master端的資料(namely data_in)。所以當valid = 1’ b1時，代表資料已送達，接著會回到state_wait_rqst這個state，然後next_notice = 1’ b0, next_ack = 1’ b1, **next_data = data_in**, next_start = 1’ b0；反之，如果當valid = 1’ b0時，會一直維持在state_wait_data這個state，next_notice = 1’ b0, next_ack = 1’ b0, next_data = data, next_start = 1'b0。



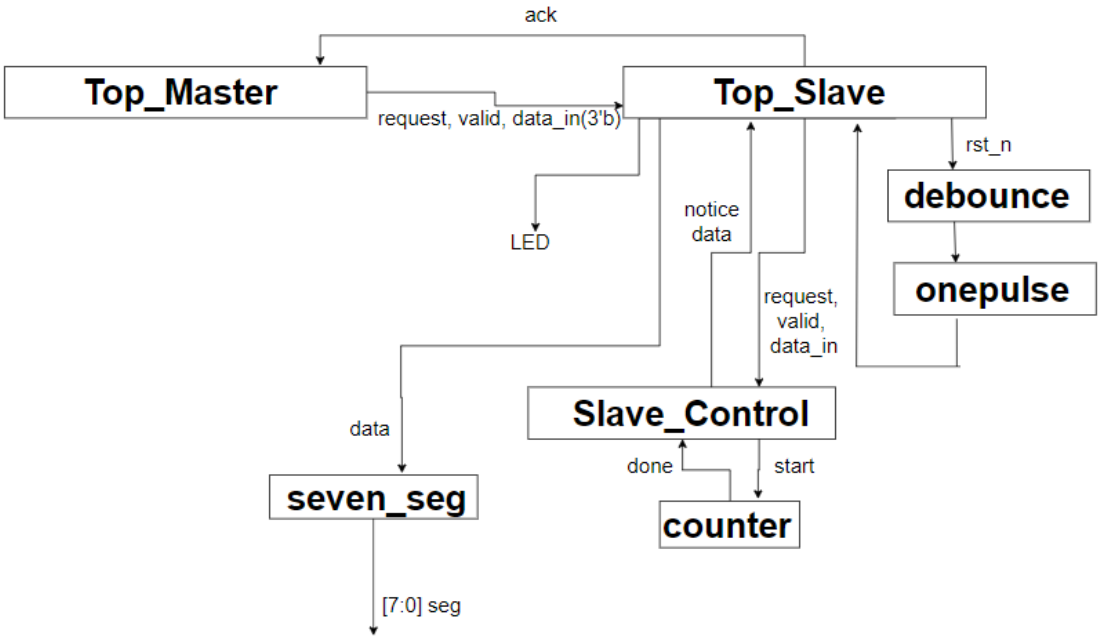
以下是完整的電路圖：



(三)FPGA Demonstration

(a)電路檢測方式：看 FPGA 板上所顯示的情形是否符合 specification。

(b)以下是此題的Block Diagram：



心得

終於完成最後一個Lab了，這次我是負責Dual FPGA communication，我發現用線FPGA互相傳輸資料很不穩定，每次我自己Demo到FPGA板上時，常常莫名其妙地跳到0這個數字，原本我以為是我寫錯，所以我de了很久，然後參考其他組的發現其實大家寫的方式都差不多，這點讓我很苦惱，後來換了線之後狀況比較好了一點，但是還是會有這樣的情況發生，不知道有沒有更好的方式可以確保兩邊傳遞的信號不會miss掉？(107060011 涂皓鈞)

心得

這個星期真的太多雜事要忙了，到星期三晚上才開始打lab，原本想說在課堂上VGA的輸出都聽懂了，這次的lab 應該是滿容易的不會花太多時間，結果真正在打的時候將概念轉換為程式碼還是需要時間的，而且需要清楚了解各個block 之間的關係還有clk trigger的時間長度，研究這個就花了一段時間，幸好最後是有把bug底出來。這次打完LAB跟結報感覺又成長了不少。(107060015 陳弘軽)

小組分工：

涂皓鈞 – 負責Dual FPGA communication 以及對應題號的結報內容 + 心得

陳弘軽 – 負責 Mixing Keyboard and VGA 以及對應題號的結報內容 + 心得